PERGAMON Computers and Mathematics with Applications 39 (2000) 187–208

An International Journal
computers & mathematics
with applications

www.elsevier.nl/locate/camwa

A Projective Simplex Algorithm Using LU Decomposition

PING-QI PAN
Department of Applied Mathematics, Southeast University
Nanjing 210096, P.R. China

(Received November 1998; accepted December 1998)

Abstract—Recently, we proposed a so-called "projective simplex method", which is amenable to linear programming problems with quite square coefficient matrix. Since it is based on QR decomposition, however, the method is not a suitable choice for large and sparse problems unless $n-m$ is far less than m, where m and n are the numbers of rows and columns of the coefficient matrix, respectively. To dodge this flaw, in this paper we propose a method using LU decomposition. In contrast to the simplex method, in which an $(m+1)\times(n+1)$ tableau is used, its tableau version handles an $(n-m)\times(n+1)$ tableau. In each iteration, its revised version solves a single $(n-m)\times(n-m)$ system only, compared with the two $m\times m$ systems solved in the revised simplex method. A complexity analysis establishes its superiority over an implementation of the simplex method in the case of the coefficient matrix being not too flat. Of particular interest might be the introduction of *deficient nonbasis* via exploiting dual degeneracy to reduce computational work further. An LU decomposition-based crash heuristic is furnished to provide "good" input. Computational results are also reported to give an insight into its interesting and distinctive behavior.

Keywords—Projection, Simplex method, Least squares problem, LU decomposition, Crash heuristic.

1. INTRODUCTION

We are concerned with the Linear Programming (LP) problem in the standard form

$$\min \quad c^{\top}x, \tag{1.1a}$$

$$\text{s.t.} \quad Ax=b, \qquad x\geq 0, \tag{1.1b}$$

where $A\in\mathcal{R}^{m\times n}$ with $m<n$, and $b\in\mathcal{R}^m$, $c\in\mathcal{R}^n$. It is assumed that c, b, and A's columns and rows are nonzero. In contrast to the conventional assumption, nothing is made on the rank of A, except for $1\leq \operatorname{rank}(A)\leq m$, although, for simplicity of exposition, the approach is derived under $\operatorname{rank}(A)=m$.

Throughout this paper, we shall denote the j^{th} column of A by a_j, and the j^{th} component of a vector $\bullet$ by $\bullet_j$. In addition, $\|\bullet\|$ designates the two-norm of a vector $\bullet$, and e_i the unit vector with the i^{th} component 1. We assume in this paper that "Gaussian elimination" means "Gaussian elimination with partial pivoting (with row interchanges)". Such doing is always possible here in

0898-1221/1999/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0898-1221(99)00323-5

our context since the portion to be triangularized will always be of full column rank. Therefore, it goes without saying that all triangular matrices encountered have nonzero diagonal entries. (See, for example, [1].)

Since its emergence in the later forties, the simplex methodology for solving LP problems has experienced a tremendous growth. There have been two trends of developments of the art. On one side, great efforts have been made to improve pivot criteria used in the simplex method to reduce the number of iterations required (e.g., [2–15]; for a survey, see [16]). On the other hand, every endeavor has also been made to reduce computational work involved in a single iteration (e.g., [17–20]).

Although it would reduce the number of iterations required, the proposed method was originally motivated by the latter. Let us take a closer look at such effort by bringing up the LU-decomposition-based simplex algorithm. Assume for the moment that $\operatorname{rank}(A) = m$. Let J_B and J_N be current basic and nonbasic index sets, respectively:

$$J_B = \{j_1, \ldots, j_m\} \qquad \text{and} \qquad J_N = \{k_1, \ldots, k_{n-m}\}. \tag{1.2}$$

Suppose that components of x and c, and columns of A are rearranged conformably:

$$\begin{aligned} x^\top &= [x_N^\top, x_B^\top] = [x_{k_1}, \ldots, x_{k_{n-m}}, x_{j_1}, \ldots, x_{j_m}], \\ c^\top &= [c_N^\top, c_B^\top] = [c_{k_1}, \ldots, c_{k_{n-m}}, c_{j_1}, \ldots, c_{j_m}], \\ A &= [N, B] = [a_{k_1}, \ldots, a_{k_{n-m}}, a_{j_1}, \ldots, a_{j_m}], \end{aligned}$$

where N and B are known as *nonbasis* and *basis* (matrix), respectively. Hereafter, we shall assume that making changes to the ordered set $\{J_N, J_B\}$ also implies rearranging components and columns of associated vectors and matrices conformably.

Construct a so-called revised tableau $[B, I]$, where $I \in \mathcal{R}^{m\times m}$ is the identity matrix. To obtain an LU factorization of B, we upper-triangularize this matrix using Gauss elimination. Let $[\hat{U}, \hat{M}]$ be the resulting tableau, so that

$$\hat{M}B = \hat{U}. \tag{1.3}$$

Using the preceding notation, we state the following version of the simplex method.

ALGORITHM 1.1. AN ITERATION OF THE LU-DECOMPOSITION-BASED SIMPLEX ALGORITHM. Assume that $\{J_N, J_B\}$ is the current index set, and that $[\hat{U}; \hat{M}]$ is the revised tableau. Given the associated basic feasible solution $\bar{x}$.

1. Compute the reduced cost
$$\bar{z}_N = c_N - N^\top \bar{y}, \tag{1.4a}$$
where
$$\bar{y} = \hat{M}^\top \hat{U}^{-\top} c_B. \tag{1.4b}$$
2. Optimality test. Stop if $\bar{z}_N \geq 0$: the solution $\bar{x}$ is optimal.
3. Select a column index q such that
$$q = \operatorname{Argmin}\{\bar{z}_{k_i} \mid i = 1, \ldots, n-m\}. \tag{1.5}$$
4. Bring k_q from J_N to the end of J_B.
5. Compute $\bar{a}_{k_q} = \hat{M} a_{k_q}$, and then
$$d_B = \begin{pmatrix} -\hat{U}^{-1}\bar{a}_{k_q} \\ 1 \end{pmatrix}. \tag{1.6}$$
6. Unboundedness test. Stop if $d_B \geq 0$: the problem is unbounded below.

7. Select a row index p and a step-length $\bar{\alpha}$ such that

$$\bar{\alpha} = -\frac{\bar{x}_{j_p}}{(d_B)_p} = \min\left\{-\frac{\bar{x}_{j_i}}{(d_B)_i} \;\middle|\; (d_B)_i < 0,\ i = 1,\ldots,m\right\}. \tag{1.7}$$

8. Update $\bar{x}$ by $\bar{x}_B := \bar{x}_B + \bar{\alpha} d_B$ along with $\bar{x}_N = 0$.
9. Bring j_p from J_B to the end of J_N.
10. Construct $[\dot{U}, \bar{a}_{k_q}; \hat{M}]$, where $\dot{U}$ results from dropping the p^{th} column of $\hat{U}$.
11. Apply Gaussian elimination to it to zero subdiagonal entries in its p through $(m-1)^{\text{th}}$ columns, and take the resulting tableau as a new $[\hat{U}; \hat{M}]$ for the next iteration.

NOTE. The search direction used in Step 8 is actually

$$d(q) = \begin{pmatrix} 0 \\ d_B \end{pmatrix} \begin{matrix} n-m-1, \\ m+1, \end{matrix} \tag{1.8}$$

which is a downhill-edge direction, satisfying

$$c^{\mathsf{T}} d(q) = \bar{z}_{k_q} < 0. \tag{1.9}$$

In geometric terms, the new vertex $\bar{x}$ minimizes the objective function c^x over the edge $\{x \mid x = \bar{x} + \alpha d(q),\ 0 \le \alpha \le \bar{\alpha}\}$, or equivalently,

$$\{x \mid Ax = b,\ x \ge 0,\ x_{k_i} = 0,\ \forall i = 1,\ldots,n-m \text{ and } i \ne q\}. \tag{1.10}$$

However, when $\bar{x}_{j_p} = 0$, and hence $\bar{\alpha} = 0$, this edge degenerates into the old vertex $\bar{x}$, and no progress will be possible in this iteration.

For existing modern implementations of the preceding algorithm, solving in some way the following two $m \times m$ triangular systems constitutes their major computations of a single iteration:

$$\hat{U}^{\mathsf{T}} \dot{y} = c_B \qquad \text{and} \qquad \hat{U}\dot{d} = -\bar{a}_{k_q}. \tag{1.11}$$

As m increases, therefore, computational work will unavoidably become more and more laborious, and may eventually break down even if $n - m$ is small (see, e.g., [21]).

To repair this shortcoming, the "projective simplex method", proposed by the author [22], solves a single $(n-m) \times (n-m)$ system only. A complexity analysis reveals a reduction of computational effort per iteration, when $n - m$ is low relative to m. Since it is based on QR decomposition, however, the method is not a very suitable choice for large and sparse problems, unless $n - m$ is far less than m. The purpose of writing this paper is to answer the following question: how can an LU-decomposition-based method be developed to be amenable to such kind of problems?

The paper is organized as follows. In the next section, we first describe the method in a tableau form. In Section 3, we recast it in a revised version using the "matrix" notation, and discuss some possible implementations of it. A complexity analysis is also made to show its superiority over the simplex method, for the case of the coefficient matrix being not too flat. In Section 4, we enhance the method by generalizing it to the dual degenerate case. Then, in Section 5, we highlight the Phase-1 issue of how to obtain a feasible solution to get the process started. Section 6 is dedicated to an LU-decomposition-based crash heuristic for providing "good" input. Finally, in Section 7, we present our computational results, offering an insight into the interesting and distinctive behavior of the proposed approach.

2. NEW SIMPLEX TABLEAU AND ASSOCIATED OPERATIONS

In this section, computations of the simplex method are rearranged in a new tableau form alternatively, without changing pivot selection criteria at all. Instead of the $(m+1) \times (n+1)$

conventional tableau, an $(n-m)\times(n+1)$ tableau is maintained and modified iteration by iteration.

Let J_B and J_N be basic and nonbasic index sets respectively, as featured by (1.2). The corresponding basic solution to (1.1) is then

$$\bar{x} = \begin{pmatrix} \bar{x}_N \\ \bar{x}_B \end{pmatrix} = \begin{pmatrix} 0 \\ B^{-1}b \end{pmatrix}. \tag{2.1}$$

In the revised simplex algorithm, this solution is however obtained in a recurrence fashion instead. The simplex multipliers y and reduced cost z_N, defined by (1.4), together with $\bar{z}_B = 0$ are the corresponding solution to (1.1)'s dual program, i.e.,

$$\max \quad b^{\mathsf{T}}y, \tag{2.2a}$$

$$\text{s.t.} \quad A^{\mathsf{T}}y + z = c, \tag{2.2b}$$

$$z \geq 0. \tag{2.2c}$$

In fact, partitioning z as $z^{\mathsf{T}} = (z_N^{\mathsf{T}}, z_B^{\mathsf{T}})$, and setting all components of z_B to zero, we can convert (2.2b) into the following $n \times n$ system:

$$\begin{bmatrix} N^{\mathsf{T}} & I_{n-m} \\ B^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} y \\ z_N \end{bmatrix} = \begin{bmatrix} c_N \\ c_B \end{bmatrix}, \tag{2.3}$$

where $I_{n-m} \in \mathcal{R}^{(n-m)\times(n-m)}$ is the identity matrix and $0 \in \mathcal{R}^{m\times(n-m)}$ is the null matrix. Such treatment of (2.2b) immediately leads to its unique solution $(\bar{y}, \bar{z})$, defined by (1.4) together with $\bar{z}_B = 0$.

It is noted that although modern implementations of Algorithm 1.1 solve in some way the $m \times m$ system $B^{\mathsf{T}}y = c_B$ for $\bar{y}$ before computing $\bar{z}_N$, what is really needed is never both $\bar{y}$ and $\bar{z}_N$, but the latter only. Taking another look at the solution of (2.2b), we can obtain $\bar{z}_N$ without having to compute $\bar{y}$ at all.

For simplicity of notation, we shall write a system in a matrix form with detached coefficients, referred to as a *tableau*. For example, (2.2b) is represented by tableau

y^{T}	z_N^{T}	z_B^{T}	RHS
A^{T}	I_N	I_B	c

(2.4)

where $I = [I_N, I_B] \in \mathcal{R}^{n\times n}$ is the identity matrix. We shall convert (2.4) into a series of tableaus of upper triangular structure using Gaussian elimination.

At first, by using appropriate Gaussian elimination, we zero entries below the diagonal of A^{T}, from its first to m^{th} columns successively, and eventually converting tableau (2.4) into

y^{T}	z_N^{T}	z_B^{T}	RHS
U_m	$\tilde{N}$	$\tilde{B}$	$\tilde{c}$
0	$\bar{N}$	$\bar{B}$	$\bar{c}$

(2.5)

where $U_m \in \mathcal{R}^{m\times m}$ is upper triangular. If $\bar{N}$ in the preceding is further triangularized similarly, then the following tableau presents:

y^{T}	z_N^{T}	z_B^{T}	RHS
U_m	$\tilde{N}$	$\tilde{B}$	$\tilde{c}$
0	U	V	$\hat{c}$

(2.6)

where $U \in \mathcal{R}^{(n-m)\times(n-m)}$ is upper triangular. As a result, the desired dual basic solution can be readily obtained from (2.6), that is,

$$\bar{y} = U_m^{-1}\left(\tilde{c} - \tilde{N}\bar{z}_N\right), \tag{2.7a}$$

$$\bar{z}_N = U^{-1}\hat{c}, \tag{2.7b}$$

$$\bar{z}_B = 0. \tag{2.7c}$$

Although it is equivalent to the solution defined by (1.4) together with $z_B = 0$ mathematically, the preceding is computationally advantageous, especially in the case when $n - m < m$, because $\bar{z}_N$ now can be computed via (2.7b) alone by solving the $(n-m) \times (n-m)$ upper triangular system below:

$$Uz_N = \hat{c} \tag{2.8}$$

via back substitution. Note that there is no need at all in the solution process for computing $\bar{y}$, the vector of simplex multipliers; in the case when an optimal dual solution is wanted, $\bar{y}$ can be covered via solving an $m \times m$ upper triangular system finally.

Assume now that the basic primal solution is feasible, i.e., $\bar{x} \geq 0$ and that $\bar{z}_N$ has been computed. If $\bar{z}_N \geq 0$, then the $\bar{x}$ and $(\bar{y}, \bar{z})$ are optimal solutions to the respective primal and dual problems, since the two solutions exhibit complementary slackness.

Let $\bar{z}_N$ be not nonnegative. Under some criteria like Dantzig's original one, determine q satisfying $\bar{z}_{k_q} < 0$. Then, what to do next is to determine a downhill-edge search direction so as the $\bar{x}$ can be updated. To this end, bring k_q from J_N to the end of J_B. Suppose that corresponding rearrangement of columns of (2.6) leads to the following tableau.

$y^\top$	$z_N^\top$	$z_B^\top$	RHS
U_m	$\tilde{N}(q)$	$\tilde{B}(q)$	$\tilde{c}$
0	$\bar{U}$	$\bar{V}$	$\ddot{c}$

(2.9)

In the preceding, the submatrix $\bar{U} \in \mathcal{R}^{(n-m)\times(n-m-1)}$, resulting from dropping the q^{th} column of U, is clearly an upper Hessenberg with nonzero subdiagonal entries in its q through $(n-m-1)^{\text{th}}$ columns. Assume that these unwanted entries are annihilated via Gaussian elimination, resulting in the following tableau.

$y^\top$	$z_N^\top$	$z_B^\top$	RHS
U_m	$\tilde{N}(q)$	$\tilde{B}(q)$	$\tilde{c}$
0	$U(q)$	$V(q)$	$\ddot{c}$

(2.10)

THEOREM 2.1. *Define the* $(m+1)$*-vector*

$$\tilde{d}_B = -\text{Sign}\left(\ddot{c}_{n-m}\right)\left(V(q)^\top e_{n-m}\right). \tag{2.11}$$

If $\tilde{d}_B \geq 0$, *then program (1.1) is unbounded below; else, the following formula is eligible for updating* $\bar{x}_B$*:*

$$\bar{x}_B := \bar{x}_B + \lambda\tilde{d}_B, \tag{2.12}$$

where

$$\lambda = -\frac{\bar{x}_{j_p}}{\left(\tilde{d}_B\right)_p} = \min\left\{-\frac{\bar{x}_{j_i}}{\left(\tilde{d}_B\right)_i} \;\middle|\; \left(\tilde{d}_B\right)_i < 0,\ i = 1, \ldots, m\right\}. \tag{2.13}$$

PROOF. It will be delayed to the end of this section.

Since components of $V(q)^\top e_{n-m})$ are available from its $(n-m)^{\text{th}}$ row, tableau (2.10) almost explicitly offers a vector $\tilde{d}_B$ needed by the formula. Suppose that $\tilde{d}_B \not\geq 0$, and a new $\bar{x}$ and

index j_p have been determined. As the component $\bar{x}_{j_p}$ now bears value zero, j_p is brought from J_B to the end of J_N. Conformably rearranging columns of tableau (2.10) renders a tableau with its first n columns upper triangularized, and hence completes a single iteration.

Nevertheless, it is observed what is really affected in the above process is only tableau's southeast portion, consisting of entries from $(m+1)$ through n^{th} rows and $(m+1)$ through $(m+n)^{\text{th}}$ columns, and the rest of the entries remain unchanged, except for the order of its columns. This is of significance because we are now faced with an $(n-m)\times(n+1)$ tableau, compared with the $(m+1)\times(n+1)$ conventional one, utilized in the simplex method. Such a tableau is referred to as a *reduced* tableau, or "tableau" for short. Specifically, that detached from (2.5), i.e.,

$$\begin{array}{cc|c} z_N^{\top} & z_B^{\top} & \text{RHS} \\ \hline \bar{N} & \bar{B} & \bar{c} \end{array} \tag{2.14}$$

is referred to as *initial* (reduced) tableau. Using notation $\bar{A} \equiv [\bar{N}, \bar{B}]$, the corresponding system can be written

$$\bar{A}z \equiv \bar{N}z_N + \bar{B}z_B = \bar{c}, \tag{2.15}$$

which is termed *the initial* (reduced) system accordingly. A tableau will always have independent nonbasic columns (corresponding to $z_N^{\top}$). If these columns are upper triangular, the tableau is said to be *canonical*; e.g., that detached from (2.6) is canonical.

$$\begin{array}{cc|c} z_N^{\top} & z_B^{\top} & \text{RHS} \\ \hline U & V & \hat{c} \end{array} \tag{2.16}$$

There are two types of tableaus (this will be generalized in Section 4): one having $n-m$ nonbasic columns, like (2.16), is said to be *regular*; the other, having $n-m-1$ nonbasic columns, is said to be *irregular*, like that detached from (2.10).

$$\begin{array}{cc|c} z_N^{\top} & z_B^{\top} & \text{RHS} \\ \hline U(q) & V(q) & \ddot{c} \end{array} \tag{2.17}$$

Summarizing preceding steps leads to the following version of the proposed method.

ALGORITHM 2.2. AN ITERATION OF PHASE-2: TABLEAU VERSION. Assume that $\{J_N, J_B\}$ is the current ordered index set, and that (2.16) is the regular canonical tableau, where $U \in \mathcal{R}^{(n-m)\times(n-m)}$ is upper triangular. Given the associated basic feasible solution $\bar{x}$.

1. Compute $\bar{z}_N = U^{-1}\hat{c}$.
2. Optimality test. Stop if $\bar{z}_N \geq 0$: the $\bar{x}$ is optimal.
3. Select subscript q, according to (1.5).
4. Bring k_q from J_N to the end of J_B, and rearrange columns of the tableau conformably.
5. Apply Gaussian elimination to the tableau to annihilate subdiagonal entries in its q through $(n-m-1)^{\text{th}}$ nonbasic columns. Let (2.17) be the resulting irregular canonical tableau.
6. Compute $\tilde{d}_B = -\text{Sign}(\ddot{c}_{n-m})(V(q)^{\top}e_{n-m})$.
7. Unboundedness test. Stop if $\tilde{d}_B \geq 0$: the program is unbounded below.
8. Determine p and λ by (2.13).
9. Update $\bar{x}$ by (2.12) along with $\bar{x}_N = 0$.
10. Bring j_p from J_B to the end of J_N, and rearrange columns of the tableau conformably.

Mathematically, the preceding algorithm is equivalent to the simplex algorithm if the same pivot rule is used. Following the same path on the underlying polyhedron, both of them proceed from vertex to adjacent vertex, while reducing objective value, until an optimal vertex is reached. So,

Algorithm 2.2 and the simplex method share common properties such as those about finiteness, outcome, and etc. On the other hand, however, since what the new algorithm deals with is an $(n-m)\times(n+1)$ rather than an $(m+1)\times(n+1)$ tableau, it should be computationally preferable in the case when $n-m<m$. (We shall return to this point, and go into details latter in Section 3.) As will be declosed latter, moreover, the present setting allows the introduction of so-called "nonbasis deficiency", which could lead to a further reduction of computational work.

So far we have described steps at the scale level. Since applying Gaussian elimination to a tableau amounts to premultiplying it by a series of Gauss transformations and permutation matrices, it is possible to recast these steps using the matrix "language". Such doing is worthwhile because it facilitates revelation of connections between quantities, as in proving of Theorem 2.1 as follows.

PROOF OF THEOREM 2.1. Assume that at some iteration we have tableau (2.10), where $U(q)\in\mathcal{R}^{(n-m)\times(n-m-1)}$ is upper triangular. Since (2.10) results from applying Gaussian elimination to (2.4), there exists a nonsingular matrix $M\in\mathcal{R}^{n\times n}$ (the product of some Gauss transformations and permutation matrices) such that

$$\begin{array}{ccc|c} y^{\top} & z_N^{\top} & z_B^{\top} & \text{RHS} \\ \hline MA^{\top} & MI_N & MI_B & Mc \end{array} \tag{2.18}$$

equals (2.10) after (2.18)'s q^{th} nonbasic column moved to the end of its basic columns. Thus, making the following change of variables:

$$x=M^{\top}x' \tag{2.19}$$

transforms the original program (1.1) into

$$\min \quad c^{\top}M^{\top}x', \tag{2.20a}$$

$$\text{s.t.} \quad AM^{\top}x'=b, \tag{2.20b}$$

$$M^{\top}x'\geq 0, \tag{2.20c}$$

whose constraints are nothing but

$$\begin{bmatrix} U_m^{\top} & 0 \\ \tilde{N}(q)^{\top} & U(q)^{\top} \\ \tilde{B}(q)^{\top} & V(q)^{\top} \end{bmatrix} x' \begin{Bmatrix} = \\ \geq \\ \geq \end{Bmatrix} \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}. \tag{2.21}$$

Consider the following least squares problem, associated with (2.10):

$$\min_{y,z_N} \left\| \begin{bmatrix} U_m & \tilde{N}(q) \\ 0 & U(q) \end{bmatrix} \cdot \begin{bmatrix} y \\ z_N \end{bmatrix} - \begin{bmatrix} \tilde{c} \\ \ddot{c} \end{bmatrix} \right\|. \tag{2.22}$$

Since the coefficient matrix of the preceding is upper triangular, the residual at the solution to it can be readily obtained, that is,

$$u=\begin{pmatrix} 0 \\ -\ddot{c}_{n-m} \end{pmatrix}. \tag{2.23}$$

It is known that u is the orthogonal projection of $-\text{dial}\,Mc$ onto the complement of the range space of the $n\times(n-1)$ coefficient matrix of (2.22), and hence a descent direction in x' space, satisfying

$$\begin{bmatrix} U_m^{\top} & 0 \\ \tilde{N}(q)^{\top} & U(q)^{\top} \end{bmatrix} u=0. \tag{2.24}$$

Of course, it is not necessary to proceed in x' space, but with the original variables x directly. From (2.19),(2.23) and equivalence of (2.18) and (2.10), we obtain the corresponding direction in x space below:

$$\hat{d}(q) = M^\top u = -\ddot{c}_{n-m} \begin{pmatrix} 0 \\ v \end{pmatrix} \begin{matrix} n-m-1, \\ m+1, \end{matrix} \tag{2.25}$$

where $v = V(q)^\top e_{n-m}$. We shall show that $\hat{d}(q)$ agrees in direction with the nonzero vector $d(q)$ defined by (1.8). From (2.24),(2.25) and the relationship between (2.18) and (2.10), it follows that $\hat{d}(q)$ satisfies $A\hat{d}(q) = 0$ and $\hat{d}(q)_N = 0$, just as $d(q)$ does. Therefore, the two vectors are in the same one-dimensional space, and hence there is some scalar β such that $\hat{d}(q) = \beta d(q)$. Then, comparing the bottom component of $\hat{d}(q)$ and of $d(q)$ yields

$$\beta = -\ddot{c}_{n-m} v_{m+1} > 0, \tag{2.26}$$

where the validity of the inequality will be shown as follows. Return the $(m+1)^{\text{th}}$ basic column to the end of nonbasic columns of tableau (2.10), resulting in a tableau equivalent to (2.6) in the sense of representing two equivalent systems (after setting $z_B = 0$). Since they have the same solution defined by (2.7), solving the equation represented by the bottom line of the resulting tableau gives the component $\bar{z}_{k_q} < 0$ of the solution, that is,

$$\bar{z}_{k_q} = \frac{\ddot{c}_{n-m}}{v_{m+1} < 0}, \tag{2.27}$$

implying the inequality in (2.26). So, $\hat{d}(q)$ is also a direction vector of the downhill edge featured by (1.10), and therefore can be used in place of $d(q)$. On the other hand, it is easy to see that $\hat{d}(q)$'s basic part (corresponding to J_B) agrees in direction with $\tilde{d}_B$, featured by (2.11). ∎

3. VARIATION: REVISED VERSIONS

It is observed that whereas the entire $(n-m) \times (n+1)$ tableau is maintained and updated in every iteration of Algorithm 2.2, not all of its entries are actually used. Such doing could be expensive, especially in the case when $n - m \ll n$. What was done in the foregoing section is only the first step toward our aim of describing the proposed approach as one could implement it as a computer program. In this section, we shall examine some possible variants of Algorithm 2.2 using matrix notation, and analyze the computational complexity of one of them, against Algorithm 1.1.

Let $\{J_N, J_B\}$ be current ordered index set and let $\bar{x}$ be the associated primal basic feasible solution. Assume that columns of the $(n-m) \times n$ coefficient matrix of the initial reduced system (2.15) has been rearranged in accordance with $\{J_N, J_B\}$, as partitioned as

$$\bar{A} = \left[\bar{N}, \bar{B}\right],$$

where the nonsingular submatrix $\bar{N} \in \mathcal{R}^{(n-m)\times(n-m)}$ is referred to as a (reduced) *nonbasis*. Suppose that the current canonical regular tableau is (2.16), resulting from the initial reduced tableau via Gaussian elimination. Let $L_1, \ldots, L_{n-m-1} \in \mathcal{R}^{(n-m)\times(n-m)}$ be Gauss transformations and $P_1, \ldots, P_{n-m-1} \in \mathcal{R}^{(n-m)\times(n-m)}$ permutation matrices such that

$$M\bar{N} = U, \tag{3.1a}$$

where

$$M = L_{n-m-1} P_{n-m-1} \ldots L_1 P_1. \tag{3.1b}$$

The preceding is referred to as the LU factorization of $\bar{N}$, and M and U as LU factors; in particular, "M" is referred to as a *left* factor. Suppose that $\bar{z}_N$ is obtained by solving the upper triangular system (2.8), and that the index k_q is chosen ($z_{k_q} < 0$), and moved from J_N to the

end of J_B (see Step 4 of Algorithm 2.2). Assume that $[\bar{N}(q), \bar{B}(q)]$ results from moving the q^{th} column of $\bar{N}$ to the end of $\bar{B}$. Then it is clear that $M\bar{N}(q)$ is nothing but the U with its q^{th} column dropped, and hence, an upper Hessenberg with nonzero subdiagonal entries in its q through $(n-m-1)^{\text{th}}$ columns. Let $\tilde{L}_q, \ldots, \tilde{L}_{n-m-1} \in \mathcal{R}^{(n-m)\times(n-m)}$ be Gauss transformations and $\tilde{P}_q, \ldots, \tilde{P}_{n-m-1} \in \mathcal{R}^{(n-m)\times(n-m)}$ permutation matrices, such that

$$\tilde{L}_{n-m-1}\tilde{P}_{n-m-1}\ldots\tilde{L}_q\tilde{P}_q\left(M\bar{N}(q)\right) = U(q), \tag{3.2}$$

where $U(q)$ is upper triangular (see Step 5 of Algorithm 2.2). Consequently, we have an LU factorization of $\bar{N}(q)$, that is,

$$\tilde{M}\bar{N}(q) = U(q), \tag{3.3a}$$

where

$$\tilde{M} = L_{n-m-1}P_{n-m-1}\ldots L_qP_qM. \tag{3.3b}$$

On the other hand, once j_p is determined, and brought from J_B to the end of J_N, the desired LU factorization of the new nonbasis $[\bar{N}(q), \bar{a}_{j_p}]$ can be readily obtained by

$$\tilde{M}\left[\bar{N}(q), \bar{a}_{j_p}\right] = \left[U(q), \tilde{M}\bar{a}_{j_p}\right], \tag{3.4}$$

where $\bar{a}_{j_p}$ is the j_p-indexed column of the initial $\bar{A}$.

The preceding formulas facilitate updating, and maintaining the LU factorization of a nonbasis, and hence enable us to recast the procedure, described in Section 2, using LU factors and the coefficient matrix $\bar{A}$ of the initial reduced system, without updating and maintaining the entire reduced tableau. For example, vector $V(q)^{\top}e_{n-m}$ in Step 6 of Algorithm 2.2 can be computed via

$$e_{n-m}^{\top}V(q) = \left(e_{n-m}^{\top}\tilde{M}\right)\bar{B}(q). \tag{3.5}$$

Since its basic and nonbasic columns are needed in this scheme, $\bar{A}$ should remain available throughout the solution process.

Alternatively, we may maintain, and update LU factors and the right-hand side in a so-called *revised* (reduced) tableau. Such an initial one can be formed as follows:

$$\left[\bar{N}; I \mid \bar{c}\right], \tag{3.6}$$

where $I \in \mathcal{R}^{(n-m)\times(n-m)}$ is the identity matrix. It is updated in subsequent iterations by premultiplying suitable Gauss transformations and permutation matrices. At any iteration, therefore, its descendant, say

$$[U; M \mid \hat{c}], \tag{3.7}$$

gives the LU factorization $M\bar{N} = U$, as well as the useful right-hand side $\hat{c}$.

We put related steps in the following model.

ALGORITHM 3.1. AN ITERATION OF PHASE-2: REVISED VERSION. Let $\bar{A}$ be the initial reduced coefficient matrix. Assume that $\{J_N, J_B\}$ is the current ordered index set, and that (3.7) is the revised tableau, where $U \in \mathcal{R}^{(n-m)\times(n-m)}$ is upper triangular. Given the associated basic feasible solution $\bar{x}$.

1. Compute $\bar{z}_N = U^{-1}\hat{c}$.
2. Optimality test. Stop if $\bar{z}_N \geq 0$: the solution $\bar{x}$ is optimal.
3. Select q according to (1.5).
4. Bring k_q from J_N to the end of J_B, and drop the q^{th} column of U.
5. Apply Gaussian elimination to the tableau to annihilate subdiagonal entries in its q through $(n-m-1)^{\text{th}}$ nonbasic columns. Denote by $[U(q); \tilde{M} \mid \ddot{c}]$ the resulting irregular canonical tableau.

6. Compute $\tilde{d}_B = -\text{Sign}(\ddot{c}_{n-m})(e_{n-m}^\top \tilde{M})\bar{B}(q)$, where $\bar{B}(q)$ consists of $\bar{A}$'s columns corresponding to J_B.
7. Unboundedness test. Stop if $\tilde{d}_B \geq 0$: the program is unbounded below.
8. Determine p and λ by (2.13).
9. Update $\bar{x}$ by (2.12) along with $\bar{x}_N = 0$.
10. Bring j_p from J_B to the end of J_N, and update the tableau by $[U; M \mid \hat{c}] := [U(q), \tilde{M}\bar{a}_{j_p}; \tilde{M} \mid \ddot{c}]$.

In the preceding algorithm, Gauss transformations and permutation matrices are accumulated in the identity matrix to maintain the left factor of an LU factorization explicitly. To exploit sparsity, however, it would be favorable to keep it as a product of these simple matrices, as has been done with the simplex method. Alternatively, the process of maintaining Gauss transformation and permutation matrices may start from scratch with the original system (2.2b) instead of the reduced system (2.15). In any case, needing to be examined are such typical issues as balancing the aim of maintaining sparsity and the aim of insisting numerical stability, appropriately ordering matrix's columns and rows to gain sparser factors, and so on. For example, a question of interest might be: how to obtain an initial reduced tableau as sparse as possible?

All these are beyond the scope of this paper, however, and will be handled separately.

It is now appropriate to take a look at the complexity of the proposed approach, against the simplex method. The latter handles an $(m+1) \times (n+1)$ tableau, while the former handles an $(n-m) \times (n+1)$ one instead. So, the new method should be favorable for solving LP problems with quite square coefficient matrix. Some improvements have been made in the reduction of computational effort per iteration also. For instance, Algorithm 1.1 solves two systems, while Algorithm 3.1 solves one only. Let us make a more precise comparison as follows. Simply counting reveals that in a single iteration, the number of multiplications and divisions required by Algorithm 1.1 is

$$N1 = nm + \left(\frac{7}{2}\right)m^2 + \left(\frac{5}{2}\right)m - 1, \tag{3.8}$$

and that by Algorithm 3.1 is

$$N2 = 3(n-m)^2 + m(n-m) + n + m. \tag{3.9}$$

Additions are about the same. Besides, Algorithms 1.1 and 3.1 require $2n + 2m - 3$ and $3n - 2$ comparisons, respectively. Consequently, we have

$$N1 - N2 = \left(\frac{9}{2}\right)m^2 - 3(n-m)^2 + \left(\frac{3}{2}\right)m - n - 1. \tag{3.10}$$

Therefore, as $m/(n-m)$ grows, the difference $N1 - N2$ increases. If, in particular, $m = n - m$, then $N1 - N2 = O((3/2)m^2)$. So, Algorithm 3.1 is more efficient than Algorithm 1.1 if the coefficient matrix is not too flat.

If the new method is generalized to the dual degenerate case, moreover, computational work required per iteration can be reduced further. This is the topic of the next section.

4. VARIATION: ALLOWING NONBASIS DEFICIENCY

Occurring very frequently in practice, degeneracy is an undesirable phenomenon both theoretically and practically. Theoretically, it undermines the applicability of finiteness theorems for almost all popular algorithms. Practically, degeneracy, although rarely leading to cycling, often causes stalling for too long a time, and consequently, degrades algorithm's performance. A bad thing might not always be bad, however. In this section, we shall show that it is possible to enhance the proposed approach by exploiting dual degeneracy.

Let us begin with redefining nonbasis.

DEFINITION 4.1. *A nonbasis is z_N's coefficient matrix which is of full column rank and whose range space includes the right-hand side of the reduced tableau.*

Nonbases can be classified into the following two categories.

DEFINITION 4.2. *A nonbasis is said to be full or deficient, if it is square or not.*

Algorithms described in previous sections involve full nonbasis only, and hence the basic feasible solution $\bar{x}$ has exactly $n-m$ nonbasic and m basic components throughout the solution process, just as in the simplex method.

Nevertheless, our present setting allows nonbasis deficiency. To explain this, assume that at some iteration we are faced with a regular canonical tableau, say (2.16), and that $\bar{z}_N (\not\geq 0)$ is degenerate; that is, some of its components are zero. Suppose without loss of generality that the last $n-m-s$ components of $\bar{z}_N$ are zero, where $0 < s < n-m$. This implies that the last $n-m-s$ components of $\hat{c}$ are zero. (It must be that $s \neq 0$ because, otherwise, optimality would have been achieved already; in this case, c is then a linear combination of rows of A, and hence the objective function $c^\top x$ is constant over the feasible region.) As a result, when we bring the last $n-m-s$ indices from J_N to J_B, and the columns of U to V correspondingly, the new U will be a deficient nonbasis, having s independent columns.

Assume now that the resulting regular canonical tableau is partitioned as

$$\begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline U & V & \hat{c} \end{array} \equiv \begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline U_1 & V_1 & \hat{c}1 \\ 0 & V_2 & 0 \end{array} \tag{4.1}$$

where $U_1 \in \mathcal{R}^{s\times s}$ is upper triangular. Assume that the associated primal solution $\bar{x}$ is nonnegative. By setting $\bar{z}_B = 0$, we obtain the corresponding dual solution from (4.1), that is,

$$\bar{z}_N = U_1^{-1}\hat{c}1. \tag{4.2}$$

Let $\bar{z}_N \in \mathcal{R}^s$ be not nonnegative. Suppose that some subscript q has been determined such that $\bar{z}_{k_q} < 0$, and index k_q has been moved from J_N and J_B. Suppose that conformably adjusting tableau (4.1) results in the following.

$$\begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline \bar{U} & \bar{V} & \bar{c} \end{array} \equiv \begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline \bar{U}_1 & \bar{V}_1 & \bar{c}1 \\ 0 & \bar{V}_2 & 0 \end{array} \tag{4.3}$$

In the case of $q = s$, the $\bar{U}_1 \in \mathcal{R}^{s\times(s-1)}$ is already upper triangular; otherwise, it is upper Hessenberg with nonzero subdiagonal entries in its q through $(s-1)^{\text{th}}$ columns. Suppose that annihilating these unwanted entries results in the following irregular canonical tableau:

$$\begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline U(q) & V(q) & \ddot{c} \end{array} \equiv \begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline U(q)_1 & V(q)_1 & \ddot{c}1 \\ 0 & \bar{V}_2 & 0 \end{array} \tag{4.4}$$

where $U(q)_1 \in \mathcal{R}^{s\times(s-1)}$ is upper triangular.

THEOREM 4.3. *Define the $(n-s+1)$-vector*

$$\tilde{d}_B = -\text{Sign}\left((\ddot{c}1)_s\right)\left(V(q)_1^\top e_s\right). \tag{4.5}$$

If $\tilde{d}_B \geq 0$, then program (1.1) is unbounded below; else, the following formula is eligible for updating $\bar{x}_B$:

$$\bar{x}_B := \bar{x}_B + \lambda\tilde{d}_B, \tag{4.6}$$

where

$$\lambda = -\frac{\bar{x}_{j_p}}{\left(\tilde{d}_B\right)_p} = \min\left\{ -\frac{\bar{x}_{j_i}}{\left(\tilde{d}_B\right)_i} \;\middle|\; \left(\tilde{d}_B\right)_i < 0,\ i = 1, \ldots, n-s+1 \right\}. \tag{4.7}$$

PROOF. It is similar to the proof of Theorem 2.1, and therefore omitted.

Suppose that $\tilde{d}_B \not\geq 0$, and a new $\bar{x}$ and index j_p have been determined. Then j_p is brought from J_B to the end of J_N, and the p^{th} basic column of the tableau is moved to the end of the nonbasic part conformably. If $s = n - m$, or else all components below the diagonal of the new end nonbasic column are zero, the resulting tableau is again regular and canonical, and hence a single iteration is completed. In the other case of $s < n - m$, these unwanted components are zeroed using Gaussian elimination in the following manner. At first, interchange rows to move the largest component (in absolute value) to the $(s+1)^{\text{th}}$ place of the column. Using this new $(s+1)^{\text{th}}$ component as a pivot, all components below it are zeroed; clearly, in such a manner the corresponding zero components of the right-hand side are not disturbed at all. If the s^{th} (diagonal) component is no less than the the $(s+1)^{\text{th}}$ component, then use it as a pivot to zero the $(s+1)^{\text{th}}$ component; otherwise, before doing so, a row interchange is conducted to move the larger $(s+1)^{\text{th}}$ component to the s^{th} place. Consequently, all components below the diagonal of the column are zero, whereas so are all ones below the $(s+1)^{\text{th}}$ component of the right-hand side. If the $(s+1)^{\text{th}}$ component of the right-hand side is zero too, then we have a new regular canonical tableau; otherwise, what we have is just a new irregular canonical tableau, and a new search direction can be determined.

From now on, the following notation will be used frequently.

$(\bullet)^{(s)}$: the submatrix or subvector, comprising the first s rows or components of a matrix or a vector $(\bullet)$.

$(\bullet)_{(s+1)}$: the submatrix or subvector, comprising the $(s+1)$ through $(n-m)^{\text{th}}$ rows or components of a matrix or a vector $(\bullet)$.

Using the preceding notation, we put these steps into the model below.

ALGORITHM 4.4. A GENERALIZATION OF ALGORITHM 2.2. Assume that $\{J_N, J_B\}$ is the current ordered set, and that (4.1) is the regular canonical tableau, where $U \in \mathcal{R}^{(n-m)\times s}$ $(s \leq n-m)$ is upper triangular. Given the feasible solution $\bar{x}$ $(\bar{x}_N = 0)$.

1. Compute $\bar{z}_N \in \mathcal{R}^s$ by solving
$$U^{(s)} z_N = \hat{c}^{(s)}.$$
2. Optimality test. Stop if $\bar{z}_N \geq 0$: the $\bar{x}$ is optimal.
3. Select q such that $q = \text{Argmin}\{\bar{z}_{k_i} \mid i = 1, \ldots, s\}$.
4. Bring k_q from J_N to the end of J_B, and adjust columns of the tableau conformably.
5. Apply Gaussian elimination to the tableau to zero subdiagonal entries in the q through $(s-1)^{\text{th}}$ nonbasic columns, resulting in an irregular canonical tableau, say (4.4).
6. Compute $\tilde{d}_B = -\text{Sign}(\ddot{c}_s)(V(q)^\top e_s)$.
7. Unboundedness test. Stop if $\tilde{d}_B \geq 0$: the problem is unbounded below.
8. Determine p and λ by (4.7).
9. Modify $\bar{x}$ by (4.6) along with $\bar{x}_N = 0$.
10. Bring j_p from J_B to the end of J_N, and adjust columns of the tableau conformably.
11. Go to step 1 if $s = n - m$.
12. Apply Gaussian elimination to the tableau to zero components below the diagonal of its newly-entering end nonbasic column.
13. Go to step 1 if the $(s+1)^{\text{th}}$ component of the right-hand side is zero.
14. Set $s := s + 1$, and go to Step 6.

NOTE 1. As s would be less than $n-m$, the new primal solution $\bar{x}$ yielded from Step 9 is feasible, but not definitely basic (see below).

NOTE 2. Manipulations of Step 12 should be done in such a manner that all corresponding zero components of the right-hand side are disturbed as less as possible, as was described prior to Algorithm 4.4.

Algorithm 4.4 is a generalization of Algorithm 2.2 to the case of dual degeneracy. Such doing would be full of promise because it somehow changes the underlying philosophy of the simplex method. Disagreeing in direction with a downhill-edge generally, the search vector, produced in an iteration where $s < n - m$, lies in the subspace parallel to a face of dimension $n - m - s + 1$, and thus searching along it no longer definitely leads to a vertex. Notably, high dual degeneracy now appears to be favorable, as it could mean a small s, and hence a high-dimensional face of such kind.

Recasting Algorithm 4.4 in the revised tableau gives the following algorithm.

ALGORITHM 4.5. A GENERALIZATION OF ALGORITHM 3.1. Let $\bar{A}$ be the initial reduced coefficient matrix. Assume that $\{J_N, J_B\}$ is the current ordered index set, and that $[U; M \mid \hat{c}]$ is the revised tableau, where $U \in \mathcal{R}^{(n-m)\times s}$ $(s \leq n - m)$ is upper triangular. Given a feasible solution $\bar{x}$ $(\bar{x}_N = 0)$.

The same as Algorithm 4.4, except for its steps replaced by the following, correspondingly.

4. Bring k_q from J_N to the end of J_B, and drop the q^{th} column of U from the revised tableau.
5. Apply Gaussian elimination to the revised tableau to zero subdiagonal entries in the q through $(s-1)^{\text{th}}$ nonbasic columns. Denote the resulting tableau by $[U(q); \tilde{M} \mid \ddot{c}]$.
6. Compute $\tilde{d}_B = -\text{Sign}(\ddot{c}_s)(e_s^\top \tilde{M})\bar{B}(q)$, where $\bar{B}(q)$ consists of $\bar{A}$'s columns corresponding to J_B.
10. Bring j_p from J_B to the end of J_N, and replace the tableau by $[U(q), \tilde{M}\bar{a}_{j_p}; \tilde{M} \mid \ddot{c}]$.
12. Apply Gaussian elimination to the tableau to zero components below the diagonal of $\tilde{M}\bar{a}_{j_p}$.

NOTES. The same as those following Algorithm 4.4.

5. PHASE-1 PROCEDURE

In order to get itself started, the proposed approach requires primal feasibility. In this section, we demonstrate that it is possible to achieve this by solving an auxiliary problem with piecewise-linear sums of infeasibilities as its objective, as have been done with the simplex method.

Let us consider such a procedure to match Algorithm 4.4.

Assume that at current iteration the solution $\bar{x}$ is infeasible with $\bar{x}_N = 0$. Define $\tilde{e} \in \mathcal{R}^{n-s}$ by

$$\tilde{e}_i = \begin{cases} 1, & \text{for } \bar{x}_{j_i} < 0, \\ 0, & \text{for } \bar{x}_{j_i} \geq 0, \end{cases} \qquad i = 1, \ldots, n - s. \tag{5.1}$$

The tactic's major point is to use the following auxiliary objective function in place of $c^\top x$:

$$-\tilde{e}^\top x_B. \tag{5.2}$$

To keep the original objective traced, nevertheless, we still manipulate the same canonical tableau as that used for Algorithm 4.4, say (4.1), where $U \in \mathcal{R}^{(n-m)\times s}$ is upper triangular. There are two cases arising as follows.

(i) $s = n - m$, or else the last $n - m - s$ components of $V\tilde{e}$ are zero. We obtain the "reduced cost", $\bar{z}_N$, by solving

$$U^{(s)} z_N = -\check{c}^{(s)}, \tag{5.3}$$

where $\check{c}^{(s)} = V^{(s)}\tilde{e}$. Thus, the right-hand side of (5.3) can be obtained by summing up all the *infeasible* basic columns of $V^{(s)}$. If $z_N \geq 0$, then it can be asserted that there exists no feasible solution to the program; otherwise, a subscript q is selected under some rule,

such that $z_{k_q} < 0$. If (4.4) is the corresponding irregular canonical tableau, the search vector is then

$$\tilde{d}_B = \text{Sign}\,(v_s\tilde{e})\,v_s^\top, \tag{5.4}$$

where $v_s = e_s^\top V(q)$.

Another point differing from Algorithm 4.4 is that the selection of p and the determination of step size λ should maintain current feasibilities.

(ii) $s < n - m$, and some of the last $n - m - s$ components of $V\tilde{e}$ are nonzero. It is easy to show that, in this case, a relevant search vector will be

$$\tilde{d}_B = V_{(s+1)}^\top \check{c}_{(s+1)}, \tag{5.5}$$

where $\check{c}_{(s+1)} = V_{(s+1)}\tilde{e}$.

We summarize the associated steps in the following model, matching Algorithm 4.4.

ALGORITHM 5.1. AN ITERATION OF PHASE-1: TABLEAU VERSION. Assume that $\{J_N, J_B\}$ is the ordered set, and that (4.1) is the canonical tableau, where $U \in \mathcal{R}^{(n-m)\times s}$ $(s \le n-m)$. Given the solution $\bar{x}$ $(\bar{x}_N = 0)$.

1. Stop if $\bar{x}_B \ge 0$: feasibility is achieved.
2. Determine $\tilde{e}$ by (5.1).
3. Go to Step 7 if $s = n - m$.
4. Compute $\check{c}_{(s+1)} = V_{(s+1)}\tilde{e}$.
5. Go to Step 7 if $\check{c}_{(s+1)} = 0$.
6. Compute $\tilde{d}_B$ by (5.5), and go to Step 10.
7. Solve system (5.3) for $\bar{z}_N$.
8. Stop if $\bar{z}_N \ge 0$: the program has no feasible solution.
9. The same as Steps 3 of Algorithm 4.4.
10. The same as Step 4 of Algorithm 4.4.
11. The same as Step 5 of Algorithm 4.4.
12. Compute $\tilde{d}_B$ by (5.4).
13. Under some rule, determine p and step-length λ such that current feasibilities are maintained.
14. The same as Step 9 of Algorithm 4.4.
15. The same as Step 10 of Algorithm 4.4.
16. The same as Step 12 of Algorithm 4.4.
17. Go to Step 1.

A Phase-1 matching Algorithm 4.5 can be created from modification of Algorithm 5.2.

ALGORITHM 5.2. AN ITERATION OF PHASE-1: REVISED VERSION. Let $\bar{A} = [\bar{N}, \bar{B}]$ be the initial reduced coefficient matrix. Assume that $\{J_N, J_B\}$ is the current ordered index set, and $[U; M \mid \hat{c}]$ the revised tableau, where $U \in \mathcal{R}^{(n-m)\times s}$ $(s \le n - m)$ is upper triangular. Given solution $\bar{x}$ $(\bar{x}_N = 0)$.

The same as Algorithm 5.1, except for its steps correspondingly replaced by these following steps.

4. Compute $\check{c}_{(s+1)} = M_{(s+1)}(\bar{B}\tilde{e})$.
7. Compute $\bar{z}_N$ by solving system (5.3), where $\check{c}^{(s)} = M^{(s)}(\bar{B}\tilde{e})$.
12. Compute $\tilde{d}_B$ by (5.4) with $v_s = (e_s^\top M)\bar{B}(q)$, where $\bar{B}(q)$ is the submatrix, corresponding to J_B, from $\bar{A}$.

As to pivot selection criteria, we used the conceptual one in the previous algorithms. More practical criteria can be used in place of them without essential difficulty in the present context. For more details, see [22].

6. LU-DECOMPOSITION-BASED CRASH HEURISTIC

In this section, we shall complete the description of our approach by offering a crash procedure for providing a reduced initial canonical tableau, together with the associated index set $\{J_N, J_B\}$. We shall show that it is possible to establish an LU-decomposition-based variant of the heuristic, proposed in [23,24].

Assume that $n \times n$ Gauss transformations $L_1, \ldots, L_m$ and permutations $P_1, \ldots, P_m$ have been determined such that

$$L_m P_m \ldots L_1 P_1 A^\top = \dot{U}, \tag{6.1}$$

where $\dot{U} \in \mathcal{R}^{n \times m}$ is upper-triangular. Let $L^{-1} = L_m P_m \ldots L_1 P_1$, and take partition

$$L^{-1} = \begin{bmatrix} H_1^\top \\ H_2^\top \end{bmatrix} \qquad \text{and} \qquad \dot{U} = \begin{bmatrix} U_m \\ 0 \end{bmatrix}, \tag{6.2}$$

where $H_1^\top \in \mathcal{R}^{m \times n}$, $H_2^\top \in \mathcal{R}^{(n-m) \times n}$, and $U_m \in \mathcal{R}^{m \times m}$. Premultiplying by L^{-1} the augmented coefficient matrix of (2.2b) leads to

$$L^{-1}\left[A^\top, I, c\right] = \left[L^{-1}A^\top, L^{-1}, L^{-1}c\right] = \begin{bmatrix} H_1^\top A^\top & H_1^\top & H_1^\top c \\ H_2^\top A^\top & H_2^\top & H_2^\top c \end{bmatrix} = \begin{bmatrix} U_m & H_1^\top & H_1^\top c \\ 0 & H_2^\top & H_2^\top c \end{bmatrix}. \tag{6.3}$$

The southeast submatrix of the preceding gives an initial reduced tableau, that is,

$$\begin{array}{c|c} z^\top & \text{RHS} \\ \hline H_2^\top & H_2^\top c \end{array} \tag{6.4}$$

Note that, corresponding to components of z, columns of $H_2^\top$ are indexed by $1, \ldots, n$, respectively. If an ordered index set $\{J_N, J_B\}$ is known already, then rearranging columns of $H_2^\top$ conformably and upper triangularizing the nonbasic columns by Gaussian elimination will produce input to Phase-1, i.e., an canonical reduced tableau. This raises the following big question: how to determine a *good* initial set $\{J_N, J_B\}$?

It is well accepted that starting from such a set that is close to an optimal one generally leads to fewer iterations required: if it happens to be optimal ideally, then no iteration is needed any more. This point should serve as the spirit of a good crash procedure. Unfortunately, existing ones, like that used in MINOS, do not make much effort along this line.

In this respect, the plausible characterization of an optimal basis [8,9] should be better than nothing at all. Bearing its essence in mind, we develop a crash heuristic, favoring an index j to be nonbasic for which the gradient, e_j, of the right-hand side of the nonnegative constraint $x_j \geq 0$ makes the most obtuse possible angle with the negative gradient, $-c$, of the objective.

The work of the determination of initial J_N and J_B is combined with the work of the upper triangularization. Assume that the initial tableau $[H_2^\top, H_2^\top c]$ is available. Initially, set J_N to empty and $J_B := \{j_1, \ldots, j_n\} \equiv \{1, \ldots, n\}$, so that all columns of $H_2^\top$ are basic. We shall select indices from J_B, one by one, to enter J_N.

Consider the least squares problem

$$\min_y \left\| L^{-1}A^\top y - L^{-1}c \right\|. \tag{6.5}$$

From (6.3), it is easy to obtain the residual at the solution to (6.5), that is,

$$r = \begin{bmatrix} 0 \\ -H_2^\top c \end{bmatrix}. \tag{6.6}$$

Clearly, $H_2^\top c = 0$ implies that c is included in the range space of $A^\top$. As this is a trivial case in which the objective value $c^\top x$ remains unaltered over the feasible region, we assume that $H_2^\top c \neq 0$. Now consider vector $\tilde{d}_B \in \mathcal{R}^{n\times n}$

$$\tilde{d}_B = -H_2 H_2^\top c. \tag{6.7}$$

The preceding is a descent direction with respect to the objective, since it holds that

$$c^\top \tilde{d}_B = -c^\top H_2 H_2^\top c = -\left\|H_2^\top c\right\|^2 < 0. \tag{6.8}$$

Moreover, from (6.2) and that the residual r is $-L^{-1}c$'s orthogonal projection onto the null space of $AL^{-\top}$, it follows that

$$A\tilde{d}_B = AL^{-\top} r = 0, \tag{6.9}$$

implying that $\tilde{d}_B$ is in the null space of A. Therefore, $\tilde{d}_B$ is eligible to be taken as a search direction. In the case when $\tilde{d}_B \geq 0$, program (1.1) either has no feasible solution or is unbounded below, because, if there is a feasible solution to it, say x, the vector $x + \lambda r$ is also feasible for an arbitrarily large $\lambda > 0$. If, otherwise, $\tilde{d}_B \ngeq 0$, then an subscript t can be determined such that

$$t = \operatorname{Arg\,min}\left\{\left(\tilde{d}_B\right)_j \mid j = 1, \ldots, n\right\}. \tag{6.10}$$

Clearly, the gradient of the right-hand side of the constraint $x_{j_t} \geq 0$ makes the most obtuse angle with $\tilde{d}_B$ among all the nonnegative constraints. Under the spirit of the plausible characterization of an optimal basis (or nonbasis), we bring j_t from J_B into J_N as its first element, and rearrange columns of (6.3) by bringing the column indexed by j_t to its first place. Consequently, its new first column corresponds to the sole nonbasic index $k_1 = t$ in J_N, and the remaining to those in J_B. Then zeroing the $n - m - 1$ entries below the diagonal of this nonbasic column using Gaussian elimination completes the first step.

Suppose now that for some s satisfying $1 \leq s \leq n - m - 1$, the s^{th} step has been finished with the presence of the following reduced canonical tableau:

$z_N^\top$	$z_B^\top$	RHS
U	V	$\hat{c}$

(6.11)

where $U \in \mathcal{R}^{(n-m)\times s}$ is upper triangular. If $\hat{c}_{(s+1)} = 0$, then reached has been a regular canonical tableau with a deficient nonbasis, U, having $s(< n-m)$ columns, and hence we are done. Assume now that $\hat{c}_{(s+1)} \neq 0$. As was with the first step, it can be shown that the nonzero vector $\tilde{d}_B \in \mathcal{R}^{n-s}$ below is relevant to be taken to determine a next basic index to enter J_N:

$$\tilde{d}_B = -V_{(s+1)}^\top \hat{c}_{(s+1)}. \tag{6.12}$$

Since what is affected in subsequent computations is the initial reduced tableau (6.4), we shall focus our attention on this part of (6.3) only. Without computing the remaining entries of (6.3), in fact, (6.4) alone can be obtained by invoking

$$H_2^\top = \left(\left(\left(I_{(m+1)} L_m\right) P_m\right) \ldots L_1\right) P_1, \tag{6.13a}$$

$$H_2^\top c = I_{(m+1)} \left(L_m \left(P_m \ldots \left(L_1 \left(P_1 c\right)\right)\right)\right), \tag{6.13b}$$

where $I_{(m+1)} \in \mathcal{R}^{(n-m)\times n}$ consists of the $(m+1)$ through n^{th} rows of the identity matrix.

Using above notation, we summarize the associated steps into the following model.

ALGORITHM 6.1. LU-DECOMPOSITION-BASED CRASH HEURISTIC. Given the coefficient matrix A and the cost vector c. This algorithm produces an initial index set $\{J_N, J_B\}$, and the associated reduced canonical tableau.

1. Compute $n \times n$ Gauss transformations $L_1, \ldots, L_m$ and permutations $P_1, \ldots, P_m$ such that
$$L_m P_m \ldots L_1 P_1 A^\top = \dot{U}$$
is upper triangular.
2. Compute $H_2^\top$ and $H_2^\top c$ via (6.13).
3. Set J_N to empty and $J_B := \{j_1, \ldots, j_n\} \equiv \{1, \ldots, n\}$.
4. Set the initial reduced tableau:
$$\begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline U & V & \hat{c} \end{array} := \begin{array}{cc|c} z_N^\top & z_B^\top & \text{RHS} \\ \hline \Phi & H_2^\top & H_2^\top c \end{array}$$
where Φ designates the empty matrix.
5. Crash steps.
For $s = 0, \ldots, n-m-1$:
(1) Stop if $\hat{c}_{(s+1)} = 0$.
(2) Compute $\tilde{d}_B \in \mathcal{R}^{n-s}$ via (6.12).
(3) Stop if $\tilde{d}_B \geq 0$: the program has no optimal solution.
(4) Select subscript t such that $t = \text{Arg}\min\{(\tilde{d}_B)_j \mid j = 1, \ldots, n-s\}$.
(5) Bring index j_t from J_B to the end of J_N, and V's corresponding column to the end of U (as its $(s+1)^{\text{th}}$ column).
(6) If $s = n-m-1$, set $s := n-m$, and stop.
(7) Set
$$[U, V \mid \hat{c}] := \tilde{L}_{s+1} \tilde{P}_{s+1} [U, V \mid \hat{c}],$$
where $\tilde{L}_{s+1}$ is a Gauss transformation and $\tilde{P}_{s+1}$ a permutation such that entries below the diagonal of the $(s+1)^{\text{th}}$ column of U is zeroed.

Clearly, the preceding algorithm produces a regular canonical tableau with a nonbasis having s columns. In particular, the termination at Step 5(1) yields a deficient nonbasis, while that at Step 5(6) offers a full one.

Finally, we point out that even though it has been assumed that $\text{rank}(A) = m$, the heuristic is able to find row-rank deficiency of A; the proposed algorithms are also suitable for the general case, almost without any modification. In the case of $r = \text{rank}(A) < m$, in fact, $\dot{U}$ will be of form
$$\dot{U} = \begin{bmatrix} U_r & U_{m-r} \\ 0 & 0 \end{bmatrix}, \tag{6.14}$$
where $U_r \in \mathcal{R}^{r \times r}$ is upper triangular, and $U_{m-r} \in \mathcal{R}^{r \times (m-r)}$. Consequently, the initial reduced tableau will be of order $(n-s) \times (n+1)$, with which the following steps of the heuristic and the two phases will get along smoothly.

7. COMPUTATIONAL RESULTS

Although a favorable justification of the proposed approach is established in Section 3, it is still interesting to see what will happen while it is put into effect. To corroborate our theory, as well as to gain an idea of the behavior of the new method, we have performed some computational trials.

The following two FORTRAN 77 codes were tested, and compared.

- Code CLS: The revised two-Phase simplex method, in which the inverse of the basis is updated and maintained explicitly.
- Code NEW: Algorithms 5.1 and 4.4 are used as Phase-1 and Phase-2, respectively. The crash heuristic, Algorithm 6.1, is furnished to provide input to them.

The preceding were coded in FORTRAN 77 models, without exploiting sparsity. Code CLS was a very efficient one available. Dantzig's original column rule and Harris' idea were implemented in both of them. Compiled using the NDP-FORTRAN-386 VER. 2.1.0. with default options; the codes were run under DOS 6.2 system on an IBM 486/66 DX2 compatible microcomputer with memory 32 Mbytes available. All the CPU time was measured in seconds with utility routine DOSTIM. The machine precision used was about 16 decimal digits. And 10^{-6} was taken to be as the primal and the dual feasibility tolerance.

The test set involves 16 standard test problems from NETLIB that do not have BOUNDS and RANGES sections in their MPS files [25], since the current version of our code cannot handle such problems implicitly. In Table 7.1, the columns labeled M and N give, respectively, the number of rows and of columns of the coefficient matrix, before adding slack variables. So, the column labeled N actually indicates the number of nonbasic columns for the simplex method ($N = n - m$). In terms of $N + M$, the test set is the largest possible subset of NETLIB problems of such type that can be solved in our computing environment.

Table 7.1. Code CLS statistics.

Problem	M	N	$M+N$	Before			After		
				Iter	Time	% Dgn	Iter	Time	% Dgn
AFIRO	27	32	59	28	0.22	71.43	29	0.22	68.97
SC50B	50	48	98	51	1.21	88.24	59	1.37	77.97
SC50A	50	48	98	51	1.15	78.43	57	1.26	71.93
ADLITTLE	56	97	153	69	2.31	24.64	128	4.07	13.28
BLEND	74	83	157	90	4.83	50.00	115	6.09	39.13
SHARE2B	96	79	175	149	12.85	67.79	196	16.48	57.14
SC105	105	103	208	110	12.25	77.27	123	13.57	70.73
STOCFOR1	117	111	228	150	20.76	80.67	174	23.73	70.11
SCAGR7	129	140	269	146	25.82	41.78	181	31.31	33.70
ISRAEL	174	142	316	188	64.04	1.06	513	159.44	0.78
SHARE1B	117	225	342	190	31.96	11.58	309	49.98	7.12
SC205	205	203	408	211	109.30	81.04	262	132.81	67.56
BEACONFD	173	262	435	159	62.34	57.86	213	81.29	51.17
LOTFI	153	308	461	190	60.53	22.63	348	106.11	13.51
BRANDY	193	249	469	238	113.25	50.00	356	163.89	34.27
E226	223	282	505	394	262.05	32.49	615	394.75	22.60
Total	1942	2412	2354	2414	784.87	46.06	3678	1186.37	31.84

Results obtained with CLS and NEW are displayed in Tables 7.1 and 7.2, respectively. The number of iterations and running time before and after a feasible solution was obtained was given in the columns under *Before* and *After*. The percentage of degenerate iterations is listed in the columns labeled *% Ndg*. The final objective values are not included, as they were achieved by both codes correctly.

Table 7.3 compares the performance of the two codes by giving iteration and time ratios of CLS to NEW. From lines labeled *Total*, it is seen that iterations required by NEW are fewer than those by CLS (with ratios 1.17 and 1.07 for "Before" and "After", respectively). The margin between the total running time required by NEW and by CLS is much larger (with ratio 2.60 for both "Before" and "After"). Note that these results were achieved even with normal problems; the ratio $N/M = 1.24$ indicates that the coefficient matrices of these test problems (after adding slack variables) are far from square. The proposed method is also more efficient than its QR-decomposition-based version [22].

Table 7.2. Code NEW statistics.

Problem	Before		After		% N1	% Dfc	% Nfl	% Dgn
	Iter	Time	Iter	Time				
AFIRO	27	0.11	29	0.11	75.00	100.00	10.34	24.14
SC50B	33	0.33	44	0.44	58.33	100.00	11.36	9.09
SC50A	33	0.38	49	0.49	58.33	100.00	12.24	16.33
ADLITTLE	106	2.09	179	3.79	84.54	100.00	15.08	1.12
BLEND	67	1.59	113	2.14	48.19	100.00	23.89	2.65
SHARE2B	129	3.41	177	4.34	83.54	100.00	39.98	22.03
SC105	61	2.80	82	3.29	56.31	100.00	6.10	30.49
STOCFOR1	60	3.74	77	4.18	43.24	100.00	15.58	11.69
SCAGR7	104	7.30	155	9.34	40.00	100.00	30.97	0.65
ISRAEL	206	17.19	461	36.36	100.00	30.59	13.88	1.52
SHARE1B	215	24.89	290	28.79	60.44	100.00	27.24	0.69
SC205	150	28.40	214	36.53	55.17	100.00	4.67	11.68
BEACONFD	124	28.12	135	29.72	45.04	100.00	4.44	2.22
LOTFI	259	55.59	559	104.30	68.83	100.00	8.23	17.53
BRANDY	126	33.29	297	47.68	42.97	100.00	11.11	3.70
E226	360	92.17	586	144.18	87.94	100.00	24.23	10.41
Total	2060	301.40	3447	455.68	62.40	90.72	16.88	8.85

Table 7.3. Comparison: Ratios of CLS to NEW.

Problem	M	N	N/M	Before		After	
				Iter	Time	Iter	Time
AFIRO	27	32	1.19	1.04	2.00	1.00	2.00
SC50B	50	48	0.96	1.55	3.67	1.34	3.11
SC50A	50	48	0.96	1.55	3.03	1.16	2.57
ADLITTLE	56	97	1.73	0.65	1.11	0.72	1.07
BLEND	74	83	1.12	1.34	3.04	1.02	2.85
SHARE2B	96	79	0.82	1.16	3.77	1.11	3.80
SC105	105	103	0.98	1.80	4.38	1.50	4.12
STOCFOR1	117	111	0.95	2.50	5.55	2.26	5.68
SCAGR7	129	140	1.09	1.40	3.54	1.17	3.35
ISRAEL	174	142	0.82	0.91	3.73	1.11	4.39
SHARE1B	117	225	1.92	0.88	1.28	1.07	1.74
SC205	205	203	0.99	1.41	3.85	1.22	3.64
BEACONFD	173	262	1.51	1.28	2.22	1.58	2.74
LOTFI	153	308	2.01	0.73	1.09	0.62	1.02
BRANDY	193	249	1.29	1.89	3.40	1.20	3.44
E226	223	282	1.26	1.09	2.84	1.05	2.74
Total	1942	2412	1.24	1.17	2.60	1.07	2.60

Such a success should partially be due to the heavy occurrence of nonbasis deficiency associated with NEW. Computational work involved in such iterations (involving a deficient nonbasis) is lower than that with full nonbasis (because in this case the system solved is of a smaller order, etc.) From the column labeled *% Dfc* in Table 7.2, giving the percentage of deficient iterations, it

is seen that with 15 out of the 16 problems, except for ISRAEL only, all iterations are such ones. The column labeled *% N1* renders the percentage of the number of the final nonbasis columns (against $N = n - m$). It shows that Code NEW terminated with a nonbasis, the number of whose columns are less than two third of $N = n - m$ overall.

Another reason why the running time, rather than iteration counts, should be taken to be as a sole index for the evaluation of codes' efficiency is as follows. It is noted that there are two types of iterations in NEW's solution process: while one (full iteration) involves both dropping from and adding into the nonbasis a column, the other involves the latter action only. From the column labeled *% Nfl*, giving the percentage of full iterations, it is seen that the fraction of such iterations are quite low (only 16.88% overall).

Table 7.4. Crash heuristic statistics.

Problem	N	Ninf	Ndinf	% NDinf	Iters	% Iters	Time	% Time
AFIRO	32	4	0	0.00	16	55.17	0.05	45.45
SC50B	48	10	0	0.00	18	40.91	0.28	63.64
SC50A	48	7	0	0.00	20	40.82	0.33	67.35
ADLITTLE	97	26	0	0.00	63	35.20	1.16	30.61
BLEND	83	20	0	0.00	33	29.20	1.16	54.21
SHARE2B	79	33	0	0.00	48	27.12	1.71	39.40
SC105	103	18	1	0.97	24	29.27	2.08	63.22
STOCFOR1	111	17	0	0.00	48	62.34	3.41	81.58
SCAGR7	140	14	6	4.29	56	36.13	5.55	59.42
ISRAEL	142	86	1	0.70	109	23.64	10.99	30.23
SHARE1B	225	48	15	6.67	128	44.14	14.67	50.96
SC205	203	46	1	0.49	56	26.17	19.01	52.04
BEACONFD	262	11	14	5.34	116	85.93	26.64	89.64
LOTFI	308	62	1	0.32	100	17.89	27.85	26.70
BRANDY	249	104	0	0.00	8	2.69	16.21	34.00
E226	282	50	27	9.57	243	41.47	50.31	34.89
Total	2412	556	66	2.74	1086	31.51	181.41	39.81

In addition, effects of degeneracy appear to be reduced a lot. From columns labeled *% Dgn* in Tables 7.1 and 7.2, giving the percentage of degenerate iterations, it is seen that overall the fraction of such iterations involved by CLS is as high as 31.84%; in contrast, that by NEW is 8.85% only.

NEW's good performance should also be due to the merit of the heuristic. In Table 7.4, the column labeled *Ninf* and *Ndinf* list the number of primal and dual infeasibilities created by the heuristic, respectively. Notably, dual feasibility was completely achieved by the heuristic alone for a half of the 16 problems. Giving the percentage of dual infeasibilities against $N = n - m$, the column labeled *% Nding* indicates that overall only 2.27% infeasibility left after its execution. On the other hand, the column labeled *% Time* shows that nearly 40% of total running time was spent on the execution of the heuristic, which might be serve as another index for NEW's high efficiency. We mention that a number of crash heuristics and pivot rules based on the plausible characterization of optimality [8,9] met similar success in other contexts. (See [10–15,23,24]).

In summary, the proposed method is very promising for solving LP problems, especially those with quite square coefficient matrix.

REFERENCES

1. G.H. Golub and C.F. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, MD, (1983).
2. P.M.J. Harris, Pivot selection methods of the Devex LP code, *Mathematical Programming* 5, 1–28, (1973).

3. D. Goldfarb and J.K. Reid, An practical steepest-edge simplex algorithm, *Mathematical Programming* **12**, 361–371, (1977).
4. S. Zionts, The criss-cross method for solving linear programming problems, *Management Science* **15** (7), 426–445, (1969).
5. T. Terlaky, A convergent criss-cross method, *Math. Oper. und Stat. ser. Optimization* **16** (5), 683–690, (1985).
6. D. Chen, P. Pardalos and M. Saunders, The simplex algorithm with a new primal and dual pivot rule, *Operations Research Letters* **16**, 121–127, (1994).
7. K. Paparrizos and K. Dosios, A new exterior point algorithm for linear programming problems, *YJOR* **4**, 137–148, (1994).
8. P.-Q. Pan, Practical finite pivoting rules for the simplex method, *OR Spektrum* **12**, 219–225, (1990).
9. P.-Q. Pan, A simplex-like method with bisection for linear programming, *Optimization* **22** (5), 717–743, (1991).
10. P.-Q. Pan, Ratio-test-free pivoting rules for a dual Phase-1 method, In *Proceedings of the Third Conference of Chinese SIAM*, (Edited by S.-T. Xiao and F. Wu), pp. 245–249, Tsinghua University Press, Beijing, (1994).
11. P.-Q. Pan, Achieving primal feasibility under the dual pivoting rule, *Journal of Information & Optimization Sciences* **15** (3), 405–413, (1994).
12. P.-Q. Pan, Composite phase-1 methods without measuring infeasibility, In *Theory of Optimization and Its Applications*, (Edited by M.-Y. Yue), pp. 359–364, Xidian University Press, Xian, (1994).
13. P.-Q. Pan, New non-monotone procedures for achieving dual feasibility, *Journal of Nanjing University, Mathematics Biquarterly* **12** (2), 155–162, (1995).
14. P.-Q. Pan, A modified bisection simplex method for linear programming, *Journal of Computational Mathematics* **14** (3), 249–255, (1996).
15. P.-Q. Pan, The most-obtuse-angle row pivot rule for achieving dual feasibility: A computational study, *European Journal of Operations Research* **101**, 167–176, (1997).
16. T. Terlaky and S. Zhang, Pivot rules for linear programming: A survey on recent theoretical developments, *Annals of Operations Research* **46**, 203–233.
17. G.B. Dantzig and W. Orchard-Hayes, The product form for the inverse in the simplex method, *Mathematical Tables and Other Aids to Computation* **8**, 64–67, (1954).
18. R.H. Bartels and G.H. Golub, The simplex method of linear programming using LU decomposition, *Communication ACM* **12**, 275–278, (1969).
19. E. Hellerman and D.C. Rarick, The partitioned preassigned pivot procedure, In *Sparse Matrices and Their Applications*, (Edited by D.J. Rose and R.A. Willouhby), pp. 68–76, Plenum Press, New York, (1972).
20. J.J.H. Forrest and J.A. Tomlin, Updated triangular factors of the basis to maintain sparsity in the product form simplex method, *Mathematical Programming* **2**, 263–278, (1972).
21. B. Hattersley and J. Wilson, A dual approach to primal degeneracy, *Mathematical Programming* **42**, 135–145, (1988).
22. P.-Q. Pan, A projective simplex method for linear programming, (1999) (to appear).
23. P.-Q. Pan, A dual projective simplex method for linear programming, *Computers Math. Applic.* **35** (6), 119–135, (1998).
24. P.-Q. Pan, A basis-deficiency-allowing variation of the simplex method for linear programming, *Computers Math. Applic.* **36** (3), 33–53, (1998).
25. D.M. Gay, Electronic mail distribution of linear programming test problems, *Mathematical Programming Society COAL Newsletter* **13**, 10–12, (1985).
26. R.H. Bartels, A stabilization of the simplex method, *Num. Math.* **16**, 414–434, (1971).
27. M. Benichou, J.M. Gauthier, G. Hentges and G. Ribiere, The efficient solution of linear programming problems—Some algorithmic techniques and computational results, *Mathematical Programming* **13**, 280–322, (1977).
28. J. Daniel, W.B. Gragg, L. Kaufman and G.W. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, *Math. Comp.* **30**, 772–795, (1976).
29. G.B. Dantzig, Maximization of a linear function of variables subject to linear inequalities, In *Activity Analysis of Production and Allocation*, (Edited by T.C. Koopmans), pp. 339–347, John Wiley & Sons, New York, (1951).
30. G.B. Dantzig, Computational algorithm of the revised simplex method, Rand Report RM-1266, The Rand Corporation, Santa Monica, CA, (1953).
31. G.B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, (1963).
32. J.J.H. Forrest and D. Goldfarb, Steepest-edge simplex algorithms for linear programming, *Mathematical Programming* **57**, 341–374, (1992).
33. P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders, Methods for modifying matrix factorization, *Math. Comp.* **28**, 505–538, (1974).
34. W. Givens, Computation of plane unitary rotations transforming a general matrix to triangular form, *SIAM J. App. Math.* **6**, 26–50, (1958).
35. G.H. Golub, Numerical methods for solving linear least squares problems, *Numer. Math.* **7**, 206–216, (1965).
36. I. Maros and G. Mitra, Simplex algorithms, In *Advances in Linear and Integer Programming*, (Edited by J. Beasley), Oxford University Press, Oxford, (1996).

37. J.L. Nazareth, *Computer Solution of Linear Programs*, Oxford Science Publications, Oxford, (1987).
38. G.W. Stewart, The economical storage of plane rotations, *Numer. Math.* **25**, 137–138, (1976).